Tool Development and Support

SPARK Toolset Release Note–Release 7.6

EXM/RN
Issue: 1.8
Status: Definitive
4th July 2008

Originator

SPARK Team

Approver

SPARK Team Line Manager

Copyright

The contents of this manual are the subject of copyright and all rights in it are reserved. The manual may not be copied, in whole or in part, without the written consent of Praxis High Integrity Systems Limited.

The software tools referred to in this manual are the subject of copyright and all rights in them are reserved. The rights in these tools are owned by Praxis High Integrity Systems Limited, and it may not be copied, in whole or in part, without the written consent of this company, except for reasonable back-up purposes. The same proprietary and copyright notices must be affixed to any permitted copies as were affixed to the original. This exception does not allow copies to be made for others, whether or not sold, and none of the material purchased may be sold, given or loaned to another person or organisation. Under law copying includes translating into another language or format.

©1991-2008 Praxis High Integrity Systems Limited, 20 Manvers St, Bath BA1 1PX

Limited Warranty

Praxis High Integrity Systems Limited save as required by law makes no warranty or representation, either express or implied, with respect to this software, its quality, performance, merchantability or fitness for a purpose. As a result, the licence to use this software is sold ‘as is’ and you, the purchaser, are assuming the entire risk as to its quality and performance.

Praxis High Integrity Systems Limited accepts no liability for direct, indirect, special or consequential damages nor any other legal liability whatsoever and howsoever arising resulting from any defect in the software or its documentation, even if advised of the possibility of such damages. In particular Praxis High Integrity Systems Limited accepts no liability for any programs or data stored or processed using Praxis High Integrity Systems Limited products, including the costs of recovering such programs or data.

SPADE is a trademark of Praxis High Integrity Systems Limited

Note: The SPARK programming language is not sponsored by or affiliated with SPARC International Inc. and is not based on SPARC™ architecture.

Contents

1 Introduction 6

2 Contact Information 7

3 SPARK language changes 8
3.1 Atomic single-field record types now permitted in RavenSPARK (SEPR 2253) 8

4 SEPR 2375 – Important Information for Users 9
4.1 Description 9
4.2 SPARK Examiner versions affected 9
4.3 Test case 9
4.4 Workaround 10
4.5 Recommended action 10

5 Examiner changes 12
5.1 Conditional updates of protected variables rejected (SEPR 2091) 12
5.2 Justification affects flow analysis (SEPR 2132) 12
5.3 Better error message for own variable declared as a constant (SEPR 2133) 12
5.4 Examiner generated RLS files may have identifiers truncated to 256 characters (SEPR 2145) 12
5.5 Constraint checks on type conversions (SEPR 2148) 13
5.6 Dynamic “for” loop where controlling variable is an “in” parameter (SEPR 2172) 13
5.7 Suppression of ‘Pos and ‘Val attributes in VCs for Integer and Character types (SEPR 2174) 13
5.8 Correction to “default_loop_assertions” warning control keyword (SEPR 2186) 13
5.9 Rejecting illegal use of subtype mark or ‘Range inside a type conversion (SEPR 2197) 13
5.10 Main subprogram inheriting package System (SEPR 2248) 14
5.11 Generation of an explicit False VC when VCG Heap exhausted (SEPR 2277) 14
5.12 Generation of VCs and Rules for T’Valid improved (SEPR 2279) 14
5.13 Output_Directory option (SEPR 2281) 14
5.14 Always generate FDL for base types (SEPR 2238) 15
5.15 Improve justification of warnings 12 and 13 (SEPR 2326) 15
5.16 RTC VCs for own variables of mode ‘in’ improved (SEPR 2287) 15

6 SPADE Simplifier 16
6.1 Simplifier abnormal termination (SEPR 2146) 16
6.2 Improved use of ground deduction rules (SEPR 2155) 16
6.3 Port to SICStus 4 and behaviour of “echo” switch (SEPR 2142) 16
6.4 Improve inequality reasoning (SEPRs 2192, 2327) 16
6.5 Improve reporting of user rules (SEPR 2242) 16
6.6 Improved efficiency with user-defined rules (SEPR 2392) 17
6.7 SPARKSimp 17

7 SPADE Proof Checker 18

8 POGS 19
8.1 Improved reporting of VCs proved false (SEPR 2149) 19
8.2 Correction to XML Output (SEPR 2153) 19
8.3 Platform independent path names when using plain option (SEPR 2164 & 2256) 19
8.4 Allow plain option with xml option to reduce cross-platform differences (SEPR 2166) 19
8.5 Improved detection and reporting of corrupt input files (SEPR 2194) 20
8.6 Improved percentages in summary table (SEPR 2240) 20

9 SPARKFormat 21
9.1 Alphabetical sorting of clauses (SEPR 2121) 21
9.2 Use of Space to separate file-names on Windows (SEPR 2223) 21
9.3 Use of alternative annotation character could produce malformed SPARK (SEPR 2225) 21
9.4 Prevent pruning of proof annotations (SEPR 2246) 21

10 SPARKMake 22
10.1 Optionally suppress generation of index and/or meta file (SEPR 2280) 22
10.2 Analyse all files if no root specified (SEPR 2282) 22

11 Backward incompatibilities 23

12 User manual change summary 24
12.1 Bookmarks in PDF user manuals (SEPR 2259) 24

13 Limitations and known errors 25
13.1 Tool limitations 25
13.2 Known error summary 27

14 Change Summary from Release 2.0 28
14.1 Release 2.0 - November 1995 28
14.2 Release 2.1 - July 1996 28
14.3 Release 2.5 - March 1997 28
14.4 Release 3.0 - September 1997 28
14.5 Release 4.0 - December 1998 29
14.6 Release 5.0 - June 2000 29
14.7 Release 6.0 - November 2001 30
14.8 Release 6.1 - June 2002 30
14.9 Release 6.3 – December 2002 32
14.10 Release 7.0 – July 2003 32
14.11 Release 7.2 – December 2004 32
14.12 Release 7.3 – February 2006 33
14.13 Release 7.31 – April 2006 34
14.14 Release 7.4 – January 2007 34
14.15 Release 7.5 – May 2007 35

15 Operating system compatibility 36

15.1 VAX/VMS 36
15.2 SPARC/Solaris 36
15.3 Windows 36
15.4 Intel/Linux 36
15.5 Apple OS X 36

Document Control and References 37
Changes history 37
Changes forecast 38
Document references 38
File under 38

1 Introduction

Continued development of the SPARK Toolset has resulted in the development of a number of useful features that have been incorporated into Toolset Release 7.6.

This document describes changes in the behaviour of all variants of the SPARK Toolset Release 7.6 compared to Release 7.5.

2 Contact Information

For further information about this document please contact Praxis High Integrity Systems:

By phone: +44 (0)1225 823829 (direct line), +44 (0)1225 466991 (exchange)

By FAX: +44 (0)1225 469006

By email: sparkinfo@praxis-his.com

3 SPARK language changes

This section describes SPARK language changes supported by Toolset Release 7.6.

3.1 Atomic single-field record types now permitted in RavenSPARK (SEPR 2253)

Previously, it was not possible to have an Atomic object of a predefined scalar type such as Boolean, Character or Integer, since the pragma must apply to a type (not an object), but these types are declared in package Standard, which is implementation-defined and cannot be changed.

To relax this restriction, the language now permits pragma Atomic to be applied to a record type under the following conditions:

- It is a user-defined non-tagged record type.
- It has exactly one field.
- That field is of a predefined scalar base type or is System.Address.

For example:

```
type Atomic_Boolean is record
   F : Boolean;
end record;
pragma Atomic (Atomic_Boolean);
```

is now accepted by the Examiner. An object of such a type can now be used to complete the declaration of a **protected** mode own variable in RavenSPARK.

Users should use representation clauses to ensure that the Size and Alignment of such a record type match the Size and Alignment of the underlying predefined type.

4 SEPR 2375 – Important Information for Users

4.1 Description

SEPR 2375 concerns a case where the Examiner fails to generate an RTC VC where the canonical semantics of Ada require a check for Constraint_Error to be performed. This means that all the remaining VCs for a subprogram can all be reported as “True” by the Simplifier, even if a potential run-time error does exist.

The exact circumstances are as follows:

- An expression includes a value type conversion that converts an expression of an enumerated type onto a subtype of that type with a narrower range.

If the variable is outside the range of the subtype then a Constraint_Error will be raised. The Examiner does not generate a VC for this check although it does generate a warning that the type conversion is not necessary. The extended form of this warning suggests that a type qualification may be used instead. If type qualification is used in place of type conversion then the warning is not generated and the Examiner does generate the appropriate VC for the constraint check.

4.2 SPARK Examiner versions affected

All Examiner versions to 7.5 inclusive are affected.

4.3 Test case

The code below illustrates the problem for a case statement:

```
package P is
   type Colour is (Red, Green, Blue, Purple);
   subtype Nice_Col is Colour range Green .. Purple;

   function F (Q : in Colour) return Nice_Col;
end P;
```

```
package body P is
   function F (Q : in Colour) return Nice_Col
   is
      R : Nice_Col;
   begin
      case Nice_Col (Q) is -- Warning 309 generated here but no VC
                           -- produced to check that Q is in range
                           -- of subtype Nice_Col. Constraint_Error
                           -- will be raised if Q is Red.
         when Green =>
            R := Green;
         when Blue | Purple =>
            R := Blue;
      end case;

      return R;
   end F;
end P;
```

4.4 Workaround

If the type conversion is replaced by a type qualification (as suggested by the extended form of the text for warning 309) then the VC for the constraint check will be generated and warning 309 will no longer be output.

Code at risk of being affected by this problem can be identified using the procedure described in section 4.5 below.

4.5 Recommended action

A full release of the toolset at version 7.6 is now available. This Examiner correctly generates the VC that was previously missing. The text for warning 309 has been modified so that the suggestion to use type qualification instead of type conversion is included in the brief version of the warning rather than just the extended form.

For example, when generating VCs for function F in the test case above, Examiner 7.6 reports:

```
  17        case Nice_Col (Q) is
                 ^
---        Warning           :309: Type conversion to own type,
consider using type qualification instead.
```

First, ensure that your standard warning control file does *not* contain the keyword "type_conversions" (or any abbreviation thereof), since this suppresses warning 309. We then recommend a complete analysis of your project using your current Examiner version. If no instances of warning 309 are reported *and* your SPARK code does not contain any *accept* annotations for warning 309, then your project is not affected by this SEPR. If warning 309 is reported then each case needs to be inspected to determine whether a narrowing conversion to a subtype of an enumeration type is being performed and, if so, whether this could lead to a run-time error. Please contact Praxis if you are unsure about how to do this.

Regardless of whether there is any potential for a run-time error, instances of warning 309 should still be checked to determine whether a type qualification would be more appropriate than a type conversion.

We recommend upgrading to Examiner release 7.6:

- firstly, to confirm that any existing analyses affected by SEPR 2375 now result in an extra VC being produced, and to check whether this can be discharged, and

- secondly, to avoid any instances of this SEPR affecting analysis of code in future.

5 Examiner changes

This section documents other significant changes to the SPARK Examiner made for release 7.6. Where appropriate, SPARK Examiner Performance Report (SEPR) numbers are given.

5.1 Conditional updates of protected variables rejected (SEPR 2091)

The restriction on the unconditional updates of protected variables has been refined so that unconditional updates of the same protected variable occurring on mutually exclusive branches are now permitted.

5.2 Justification affects flow analysis (SEPR 2132)

In some circumstances, when using records, the addition of an accept statement could affect the results of flow analysis. This anomaly has been resolved.

5.3 Better error message for own variable declared as a constant (SEPR 2133)

A common mistake in SPARK is to attempt to complete the declaration of an own variable with a constant declaration. Previously, the Examiner issue semantic error 10 (“illegal redeclaration”). This case is now detected specifically, and new more informative semantic error 12 is issued. For example:

```
package P
--# own S;
is
   S : constant Boolean := True;
   ^1
*** (  2)  Semantic Error    : 12: Own variable S can only be
           completed by a variable declaration, not a constant
           [Explanatory note: If the object in question is really
           a constant, then remove it from the enclosing package's
           own variable annotation.].
end P;
```

5.4 Examiner generated RLS files may have identifiers truncated to 256 characters (SEPR 2145)

The Examiner no longer truncates identifiers in RLS files to 256 characters. This limit has been raised to 1024 and if this new limit is exceeded the Examiner will terminate with a fatal error rather than silently truncating the text.

5.5 Constraint checks on type conversions (SEPR 2148)

The Examiner now performs a constraint check on type conversions within static expressions.

5.6 Dynamic “for” loop where controlling variable is an “in” parameter (SEPR 2172)

If a “for” loop has a dynamic range where one or more of the controlling variables is an “in” parameter, then that variable cannot be changed in the body of the loop. Therefore, the Examiner does not need to synthesize an “X%” variant of the variable for use in VC Generation. In such cases, an explicit assertion “X = X%” is no longer required in the loop body.

5.7 Suppression of ‘Pos and ‘Val attributes in VCs for Integer and Character types (SEPR 2174)

The Ada LRM states that the position value of an integer typed value (signed or modular) is equal to that value. Furthermore, in SPARK, type Character is modelled in FDL as integer, so it is always the case that Character’Pos (X) = Character’Val (X) = X.

The Examiner now applies this simplification during VC generation, resulting in smaller, simpler VCs. The rules for the replacement of Character’Pos (X) and Character’Val (X) are no longer required, so these are no longer generated in the RLS file.

5.8 Correction to “default_loop_assertions” warning control keyword (SEPR 2186)

The keyword required to control the appearance of the warning for a default loop assertion when generating VCs is “default_loop_assertions.” This was given incorrectly in the Examiner User Manual, warning explanation and the errors.htm file, and has been corrected.

5.9 Rejecting illegal use of subtype mark or ‘Range inside a type conversion (SEPR 2197)

Previously, the Examiner could silently accept the illegal use of a subtype mark or a ‘Range attribute as the argument of a value type conversion. This could lead to mal-formed VCs being generated. This case is now correctly rejected by the Examiner.

5.10 Main subprogram inheriting package System (SEPR 2248)

Previously, the Examiner did not permit package System to be inherited by the main subprogram, assuming package System is declared in the configuration file. This has been corrected.

5.11 Generation of an explicit False VC when VCG Heap exhausted (SEPR 2277)

Previously, if the VC Generator Heap became exhausted, the Examiner terminated leaving a possibly empty VCG file.

This behaviour has now been changed. Specifically, in the event of a VCG Heap overflow, the Examiner now

1 Generates a single “False” VC for the offending program unit. This will show up in subsequent analysis with POGS. The VCG file also contains a comment “False VC generated due to VCG heap exhausted” that will be carried through by the Simplifier into a resulting simplified VCs.

2 Generates a new warning (409) for the offending subprogram. This warning reads “VCs could not be generated for this subprogram due to its size and/or complexity exceeding the capacity of the VC Generator. Unprovable (False) VC generated.”

3 Carries on with further analysis with the next program unit in the enclosing compilation unit.

5.12 Generation of VCs and Rules for T’Valid improved (SEPR 2279)

Ada LRM 13.9.1(2) defined a valid value as being a member of the indicated subtype for an object. This is now modelled more precisely by the VC Generator, modelling ‘Valid in terms of the subtype rather than the underlying base-type.

This change particularly improves the effectiveness of exception freedom proof for code that reads values of a particular subtype from an external variable.

5.13 Output_Directory option (SEPR 2281)

The Examiner now has a command-line option “output_directory” that can be used to control the directory where the Examiner’s report file, listing files, VCs and/or Path Functions are generated. This is particularly useful where source code appears “read-only” in a directory where files cannot be created.

See the Examiner User Manual sections 3.1 and 4.9 for more details.

5.14 Always generate FDL for base types (SEPR 2238)

Previously, the Examiner may have generated an RLS file that contained base type constants not declared in the FDL. The missing FDL information had the potential to affect the application of the Simplifier and the Checker. Now, FDL is generated for all base type constants that are used by a subprogram.

5.15 Improve justification of warnings 12 and 13 (SEPR 2326)

Warnings 12 and 13 relate to the use of unchecked conversions. Previously, the Examiner would reject justifications for these warnings if the instantiation of the unchecked conversion function was not in the same package as the use being warned about. This has now been corrected.

5.16 RTC VCs for own variables of mode ‘in’ improved (SEPR 2287)

Previous versions of the Examiner have made the simple but rather blunt assumption that if a subprogram imports an external (**in** mode own) variable then local variables of the subprogram may contain invalid values. Local variables of such a subprogram are not assumed to be in-type, potentially reducing the possibility for an automatic proof by the simplifier, particularly if an assert statement is inserted.

This simple assumption is safe but can be too general—all subprograms that import an external variable (either directly or indirectly) were affected. Examiner 7.6 now only considers the subprograms that directly read from an external variable as having local variables with potentially invalid values.

The new approach is justifiable because all the exported variables of a subprogram are checked when the subprogram is analysed to ensure that they are not derived from a possibly invalid value. A warning is also generated by the Examiner when an external variable is read to ensure the programmer and reviewer are aware of the possibility of an invalid value. The same approach is already in use by the Examiner when a subprogram calls an unchecked conversion. For more information see the “Generation of RTCs for SPARK Programs” manual section 4.2.

6 SPADE Simplifier

Toolset release 7.6 ships with Simplifier version 2.39 and includes the following improvements:

6.1 Simplifier abnormal termination (SEPR 2146)

The Simplifier includes improved constraints to avoid non-terminating recursion when using non-ground rules involving enumerated types.

6.2 Improved use of ground deduction rules (SEPR 2155)

The use of ground (fully instantiated) deduction rules within the Simplifier has been improved.

Such rules are now added as additional hypotheses just prior to the “Proof Framing” phase. The additional hypotheses may then be exploited by later proof phases and to satisfy any side conditions needed by user defined proof rules.

6.3 Port to SICStus 4 and behaviour of “echo” switch (SEPR 2142)

The Simplifier has been ported to SICStus 4. As a side-effect of this change, the /memory switch has been removed (already deprecated since release 7.3). In addition the /noecho switch has been removed, and by default the behaviour is as it was previously when invoked with /noecho. If desired, a new switch “/echo” will reproduce the old behaviour.

6.4 Improve inequality reasoning (SEPRs 2192, 2327)

Previously, in reasoning about a specific combination of integer inequalities, the Simplifier could enter a non-terminating loop, eventually failing due to lack of resources. Additional guards have been introduced to avoid such non-terminating loops. (SEPR 2192)

Secondly, the Simplifier is more powerful in reasoning about integer inequalities that involve a proof by transitivity. Such cases can arise from common forms of “for” loop in SPARK programs. (SEPR 2327)

6.5 Improve reporting of user rules (SEPR 2242)

Previously, where applying multiple user rules with similar conditional parts, the Simplifier may not report the application of some user rules. These user rules are now reported.

6.6 Improved efficiency with user-defined rules (SEPR 2392)

The Simplifier now gives priority to used-defined proof rules that match a conclusion and result in a fully-instantiated list of Conditions. Users are recommended to avoid unnecessary unbound variables in Condition lists, since these can affect Simplifier performance. See section 7 of the Simplifier User Manual for more details.

6.7 SPARKSimp

SPARKSimp v3.2 ships with toolset release 7.6. This version incorporates a single correction – the titles of the columns in SPARKSimp's progress log are now correct – the "Thread" number is never reported, so this title has been removed (SEPR 2278).

7 SPADE Proof Checker

Proof Checker version 2.11 ships with toolset release 7.6. This version corrects a single issue regarding the behaviour of the Checker when the FLEXlm “TimeOut” feature has expired (SEPR 2276).

8 POGS

Toolset 7.6 ships with POGS version 5.0 which includes the improvements covered in the following subsections.

8.1 Improved reporting of VCs proved false (SEPR 2149)

POGS has an improved format for summarising the VCs proved false and a separate summary reporting the highest priority proof tool used in proving a subprogram in the order Examiner -> Simplifier -> User-Defined Proof Rules -> Checker -> By Review.

POGS now checks the invariant that the number of subprograms fully proven plus the number of subprograms which have a VC proved False plus the number of subprograms with at least one undischarged VC but with no VCs proved False must equal the number of subprograms for which VCs have been generated. If POGS fails the check it terminates with a fatal error.

The new format and the invariant check are described in the updated POGS User Manual.

8.2 Correction to XML Output (SEPR 2153)

The use of user defined proof functions was reported as plain text in the XML output. This has been resolved but note that the reporting of the use of user defined proof rules is not included as part of the XML output.

8.3 Platform independent path names when using plain option (SEPR 2164 & 2256)

When the /p (or –p on Unix systems) option is selected the path names appear with "/" separators independent of the operating system on which POGS is run. This change assists in comparing POGS summary reports produced on different operating systems. See the updated POGS User Manual for details.

8.4 Allow plain option with xml option to reduce cross-platform differences (SEPR 2166)

The /p (or –p on Unix systems) option can now be selected in conjunction with the /x option. This replaces the current date and time with a standard message in the XML file and normalises the start directory name. The changes simplify cross checking results from multiple test runs and test runs from different operating systems.

8.5 Improved detection and reporting of corrupt input files (SEPR 2194)

Previously, POGS would not terminate if encountering an empty siv file. Further, where encountering other classes of corrupt files, POGS may have displayed a malformed results table. Now, common classes of corrupt files will be detected and reported to the screen, the summary file, and in the overall summary results. Where corrupt files are detected their associated results table is suppressed. For more details see the updated POGS user manual.

8.6 Improved percentages in summary table (SEPR 2240)

Previously, due to rounding errors, POGS may have reported 0% for a small non-zero value or 100% for a large value not equal to the total. Now, POGS treats 0% and 100% specially so that they always mean exactly 0% and exactly 100% respectively. Further, “<1%” is shown where the value is close to 0% and “>99%” is shown where the value is close to 100%. For more details see the updated POGS user manual.

9 SPARKFormat

Toolset 7.6 ships with SPARKFormat version 7.6 which has the following improvements:

9.1 Alphabetical sorting of clauses (SEPR 2121)

A new option "/order" has been added to SPARKFormat to facilitate the alphabetic ordering of clauses. See the updated SPARKFormat User Manual.

9.2 Use of Space to separate file-names on Windows (SEPR 2223)

SPARKFormat on Windows now permits the use of white-space to separate multiple file-names on the command-line. As a result of this, wild-cards such as "sparkformat *.ads" now work as expected on Windows.

9.3 Use of alternative annotation character could produce malformed SPARK (SEPR 2225)

Previously, in certain situations involving longer lines, where an annotation character other than '#' is used, the reformatted annotations may become malformed. This flawed behaviour has been corrected.

9.4 Prevent pruning of proof annotations (SEPR 2246)

Previously, proof annotations immediately following global annotations would be silently deleted. This flawed behaviour has been corrected.

10 SPARKMake

SPARKMake release 7.6 ships with toolset release 7.6 which has the following improvements:

10.1 Optionally suppress generation of index and/or meta file (SEPR 2280)

New options "/noindexfile" and "/nometafile" have been added. These options suppress the generation of the index and meta files respectively. See the SPARKMake User Manual for further information.

10.2 Analyse all files if no root specified (SEPR 2282)

If no root file is specified SPARKMake will produce a meta file for the analysis of all files that it encounters (provided that there is an appropriate order in which they can be analysed).

SPARKMake also now informs the user of any SPARK source files it encountered but which it could not include in the meta file (for example because of circular dependencies).

See the SPARKMake User Manual for further information.

11 Backward incompatibilities

There are no known backward incompatibilities introduced between Examiner Releases 7.5 and 7.6.

12 User manual change summary

The following user manuals have been changed between Examiner Releases 7.5 and 7.6.

- RavenSPARK language definition and RavenSPARK Rationale– updated to allow Atomic single-field record types.
- Examiner User Manual – updated for new and improved error messages.
- POGS User Manual.
- SPARKFormat – new “order” command line switch.
- SPARKMake User Manual – new “noindexfile” and “nometafile” switches; new functionality to analyse all files if no root is specified.
- Generation of RTCs for SPARK Programs – updated to describe improved VC generation in the presence of external own variables.

12.1 Bookmarks in PDF user manuals (SEPR 2259)

The PDF versions of the user manuals now include PDF bookmarks for ease of navigation.

13 Limitations and known errors

13.1 Tool limitations

This section describes limitations of the Examiner tool arising mainly from incomplete implementation of planned features. Where appropriate a SPARK Examiner Performance Report (SEPR) number is given.

13.1.1 General

1 The SPARK 95 language definition removes the distinction between initial and later declarative items; this distinction remains in force in the Examiner that requires SPARK 83 declaration orders even in SPARK 95 mode. (SEPR 813)

2 The Examiner does not yet permit the use of 8-bit characters in SPARK 95 user-defined identifiers. (SEPR 818)

3 Universal expressions in a modular context may sometimes require type qualification. (SEPR 1591)

4 The Examiner does not yet permit the use of “use type” following an embedded package specification. (SEPR 747)

5 The Examiner does not yet permit the renaming of packages in the same way that subprograms can be renamed. (SEPR 1391)

6 The Examiner does not yet allow the ‘Base attribute when not used as a prefix. (SEPR 1114)

7 The Examiner does not yet allow S’Range where S is scalar. (SEPR 1115)

13.1.2 Verification Condition Generation and Run-time Checks

1 Ada string inequality is not modelled. (SEPR 712)

2 VCs involving string catenation that includes the character ASCII.NUL will be incorrect. (SEPR 661)

3 Aggregates of multi-dimensional arrays cannot be modelled although aggregates of arrays of arrays can. (SEPR 590)

4 Verification conditions involving real numbers are evaluated using infinite precision or perfect arithmetic; this allows the correctness of an algorithm to be shown but cannot guard against the effects of cumulative rounding errors for example.

5 The Examiner does not generate VCs for package initialization parts. Statically determinable constraint errors will be detected during well-formation checking of package initialization. (SEPR 288)

6 The VC Generator cannot model the implementation-dependent attributes of floating and fixed-point types; see section 13.1.3.

13.1.3 Attribute limitations

13.1.3.1 Unimplemented attributes

The following attributes are officially supported by SPARK according to the language definition, but are not yet implemented by the Examiner. The Examiner will generate error number 30 ("Attribute XXX is not yet implemented in the Examiner") if you try to use them.

- Adjacent
- Compose
- Copy_Sign
- Leading_Part
- Remainder
- Scaling
- Exponent
- Fraction
- Machine
- Model
- Rounding
- Truncation
- Unbiased_Rounding

Note that these are all function-like attributes concerning floating- and fixed-point types.

13.1.3.2 Unevaluable attributes

The Machine_* and Model_* attributes are accepted by the Examiner, but it does not know how to statically evaluate them since they are inherently implementation dependent. For example, the package:

```
package F is
```

```
   type T is digits 6 range –10.0 .. 10.0;
   C : constant := T'Machine_Emax;
end F;
```

is legal SPARK, but the Examiner does not know the actual numeric value of C.

13.2 Known error summary

This section lists known errors in the Examiner that are awaiting investigation and correction.

1 The SPARK rule that array actual parameters must have the same bounds as the formal parameter is not checked for function parameters where the actual parameter is a subtype of an unconstrained array type. Since subtype bounds are static in SPARK errors of this kind should be detected by an Ada compiler. If not an unconditional run-time error will occur. (SEPR 1060)

2 The Examiner permits the body of a subprogram to be entirely made up of proof statements thus breaching the Ada rule that at least one Ada statement must be present. (SEPR 278)

3 Where a package declares two or more private types the Examiner permits mutual recursion between their definitions in the private part of the package. (SEPR 848)

4 The Examiner does not take due account of a range constraint when determining the subtype of a loop variable; this affects completeness checking of case statements within the loop. For example `for I in Integer range 1..4 loop` would require only values 1, 2, 3 and 4 to be covered by the case statement. (SEPR 693)

5 When summarising the counts of pragmas found during an analysis the totals may depend on whether units are selected via the command line (or metafile) or using the index mechanism. The difference affects only pragmas placed *between* program units and arises because placing a file name on the command line causes the entire *file* to be analysed whereas selecting it using indexes causes only the required *unit* to be read from the file. (SEPR 483)

14 Change Summary from Release 2.0

A release note detailing changes from the previous version accompanies each Examiner Release; this section simply summarises the various changes that have been made.

14.1 Release 2.0 - November 1995

Release 2.0 added:

- static expression evaluation;
- variable initialization at declaration;
- full-range scalar subtypes; and
- operator renaming in package specifications.

14.2 Release 2.1 - July 1996

Release 2.1 added:

- facilities for proof of absence of run-time errors

14.3 Release 2.5 - March 1997

Release 2.5 was distributed with “High Integrity Ada - the SPARK Approach” and provided initial facilities for SPARK 95

14.4 Release 3.0 - September 1997

Windows NT was supported for the first time with this release. Release 3.0 also added:

- additional SPARK 95 support;
- flow analysis of record fields;
- command line meta files;
- named numbers;
- unqualified string literals;
- moded global annotations; and

- optional information flow analysis.

14.5 Release 4.0 - December 1998

With Release 4.0 we upgraded all users to a single product standard supporting SPARK 83, SPARK 95 and analysis options up to an including proof facilities. New features were:

- full implementation of public and private child packages;
- default switch file; and
- provision of the INFORMED design document.

14.6 Release 5.0 - June 2000

- Enhanced proof support:
  I. facilities for proof of programs containing “abstract state”;
  II. addition of quantified expressions;
  III. proof rule generation for enumeration types;
  IV. identification of the kind and source of each VC;
  V. suppression of trivially true VCs;
  VI. Proof Obligation Summariser tool (POGS)
- Optional HTML output files with hyperlinks that can be “browsed” interactively
- Better support for common Ada file naming conventions
- User-selectable annotation character
- Improved suppression of analysis were results might otherwise be misleading
- Static expression evaluation in proof contexts
- Singleton enumeration types
- Revised SPARK_IO package
- Error numbering

14.7 Release 6.0 - November 2001

- Introduction of “external variables” to simplify modelling of the interactions between a SPARK program and its external environment.
- Addition of the “null derives” annotation to describe information flows which affect only the external environment.
- Introduction of modular types
- Use of loop labels in exit statements
- Use of global modes on function subprograms
- Extended support for predefined types such as Long_Integer
- Simplified run-time check generation for own variables
- Relaxation of need for mandatory type announcement of own variables
- Plain output option to simplify comparisons of Examiner output files
- Platform-independent switch files and metafiles
- Support for intentionally infinite loops
- Detection of own variables that can never be initialized
- Detection of unusable private types
- Extra refinement checks on global variables when performing data flow analysis
- Detection of unnecessary others clause in case statements
- Extensions to the POGS tool
- Improved error messages to distinguish different cases of variables which are “not declared or visible”
- Improved SPADE Simplifier Release 2.0
- New “SPARKSimp” Tool

14.8 Release 6.1 - June 2002

- Introduction of tagged types

- Introduction of type assertion annotations
- Introduction of modular subtypes
- Introduction of the configuration file
- Introduction of the `help` command line switch
- Demo Examiner now runs on Linux.
- VCG generation for inherited operations of tagged types
- Improved handling of null derives
- Attributes ‘Floor and ‘Ceiling implemented
- Detection of duplicate record fields
- Improved overflow checks on universal integer expressions
- Corrected handling of loop invariants in while loops
- Strengthened behaviour of /noecho option
- Trapping non-positive accuracy in real type declaration
- Recursion in meta-files and index-files
- Improved handling of Address clauses
- Improved handling of Import and Interface pragmas
- VCG Modelling of Boolean membership operators
- Simplification of common Integer inequalities
- Simplification of common enumerated inequalities
- Simplification of VCs involving quantified expressions
- Simplifier performance
- Checker has new built-in rule families: MODULAR, BITWISE and ENUMERATION
- Proved by review option in POGS

14.9 Release 6.3 – December 2002

Release 6.3 of the toolset was constructed to accompany the textbook "High Integrity Software: The SPARK Approach to Safety and Security" by John Barnes, but was not delivered to other users. Its main new features were:

- Slight revision to the rules regarding the placement of tagged type declarations.
- Correction to the modelling of Boolean type membership operators in the verification conditions.
- Support for generating VCs that allow the verification of the Liskov Substitution Principal (LSP) for tagged types and their operations.
- Dramatically improved performance of the Simplifier, particularly in the simplification of quantified expressions.

14.10 Release 7.0 – July 2003

Release 7.0 of the toolset comprised:

Examiner version 7.0

Simplifier version 2.12

Release 7.0 added:

- Ravenscar profile extensions to the language.
- Support for Ada.Interrupts and Ada.Real_Time in the configuration file.
- The new /noduration command line switch.
- VC generation for unconstrained formal parameters.
- Suppression of VC generation for illegal function bodies.
- New "SPARKFormat" tool.

14.11 Release 7.2 – December 2004

Release 7.2 of the toolset comprised:

Examiner version 7.2

Simplifier version 2.17

Release 7.2 added:

- Unconstrained string constants to the language.
- Instantiation of Unchecked_Conversion to the language.
- (Full) record subtypes to the language.
- Declaration of subprograms in the private part of packages.
- Refined proof annotations for private types.
- % suffix for referring to value of variable on entry to loop in proof contexts.
- Extra hypotheses for local variables.
- Suppression of VC generation for illegal function bodies.
- Replacement rules for composite constants.
- Concurrent simplification with SPARKSimp.
- Improved simplification of VCs with large structured objects.
- Improved simplification of arithmetic and logical expressions.
- New “SPARKMake” tool.

14.12 Release 7.3 – February 2006

Release 7.3 of the toolset comprised:

Examiner version 7.3

Simplifier 2.22

Checker 2.06

Significant features of this release included:

- Improved VC Generation.
- New “error explanations” switch for Examiner.
- Generation of proof rules for ‘Size.
- Improved diagnosis and reporting of common syntax errors.

- Error and warning count summary in Examiner output.
- Use of pragma Import to complete an external own variable.
- Correction to FDL declaration order for private and announced types.
- New Simplifier tactics for dealing with rational inequalities and Examiner-generated proof rules.
- User-defined proof rules for the Simplifier.
- New Checker rules for MODULAR expressions.
- Significant performance improvement for Simplifier and Checker.

14.13 Release 7.31 – April 2006

Release 7.31 of the toolset comprised:

Examiner version 7.31

Simplifier 2.24

Checker 2.07

Significant features of this release included:

- Correction to flow analysis of array element “out” parameters.
- Port of “Demo” toolset to Mac OS X.
- New /typecheck option in Simplifier.
- Improved validity checking of user-defined proof rules in the Simplifier.
- Correction to ARITH proof rules in the Checker.

14.14 Release 7.4 – January 2007

Release 7.4 of the toolset comprised:

Examiner version 7.4

Simplifier 2.30

Checker 2.08

Significant features of this release included:

- The ‘accept’ annotation.
- Conditional flow errors are now reported as errors.
- The Simplifier supports user defined proof rules.
- SPARKFormat supports a wider range of annotations.
- SPARKMake has the ability to generate relative pathnames.

14.15 Release 7.5 – May 2007

Release 7.5 of the toolset comprised:

Examiner version 7.5

Simplifier 2.32

Checker 2.08

Significant features of this release included:

- Correct VC generation for record fields and array elements used as “in” or “in out” parameters.

15 Operating system compatibility

15.1 VAX/VMS

The Examiner (but not any of the other tools) is supported on VAX/VMS versions 5.5-2 or higher.

15.2 SPARC/Solaris

The toolset is compatible with Solaris 8 through to 10 including those with a 64-bit kernel.

n.b. Solaris 6 and 7 are no longer supported.

15.3 Windows

The toolset is compatible with Windows NT 4.0, 2000, XP and Vista.

15.4 Intel/Linux

All the toolset, with the exception of the Checker, is compatible with Intel-based Linux operating systems, in both 32-bit and 64-bit versions. Only the “FreeDemo” version of the toolset is currently available for Linux to support buyers of John Barnes’ “SPARK Book.” If you require a full professional SPARK toolset for Linux, then please contact us.

15.5 Apple OS X

All the toolset, with the exception of the Checker, is compatible with Apple’s Intel-based OS X/Darwin operating systems. The toolset is built and tested on OS X Server version 10.4.10. Only the “FreeDemo” version of the toolset is currently available for OS X to support buyers of John Barnes’ “SPARK Book.” If you require a full professional SPARK toolset for OS X, then please contact us.

Document Control and References

Praxis High Integrity Systems Limited, 20 Manvers Street, Bath BA1 1PX, UK.
Copyright © Praxis High Integrity Systems Limited 2008. All rights reserved.

Changes history

Issue 0.1 (2nd August 2007):	First draft for release 7.5d01
Issue 0.2 (15th August 2007):	Second draft for release 7.5d01
Issue 0.3 (17th August 2007):	Release 7.5d02 upgrade to GNAT Pro 6.0.2
Issue 0.4 (21st August 2007):	Release 7.5d03 upgrade to Sicstus 4.0.1
Issue 0.5 (31st August 2007):	Add description of SEPR 2174.
Issue 0.6 (25th January 2008):	Add description of SEPR 2253.
Issue 0.7 (29th January 2008):	Updated for release 7.5d03 following review.
Issue 0.8 (18th February 2008):	Updated for SPARKSimp version 3.2.
Issue 0.9 (19th February 2008):	Updated for SEPR 2277.
Issue 1.0 (20th February 2008):	Updated for Simplifier 2.37 and Checker 2.10 (SEPR 2276).
Issue 1.1 (25th February 2008):	Updates for release 7.5d03 following review.
Issue 1.2 (1st April 2008):	Updates for Examiner output_directory switch (SEPR 2281), SPARKMake, POGS.
Issue 1.3 (11th April 2008):	Updates for warnings 12 and 13 justification (SEPR 2326), and Simplifier inequality reasoning (SEPR 2327), prior to 7.6.
Issue 1.4 (17th April 2008):	Updates for PDF bookmarks in documentation (SEPR 2259).
Issue 1.5 (8th May 2008):	Updated for SEPR 2287.
Issue 1.6 (22nd May 2008):	Updated for SEPR 2375.
Issue 1.7 (11th June 2008):	Definitive issue for release 7.6 following review.
Issue 1.8 (4th July 2008):	Addition of Simplifier 2.39 following SEPR 2392.

Changes forecast

None.

Document references

None.

File under

SVN:trunk/userdocs/Examiner_RN_7p6.doc